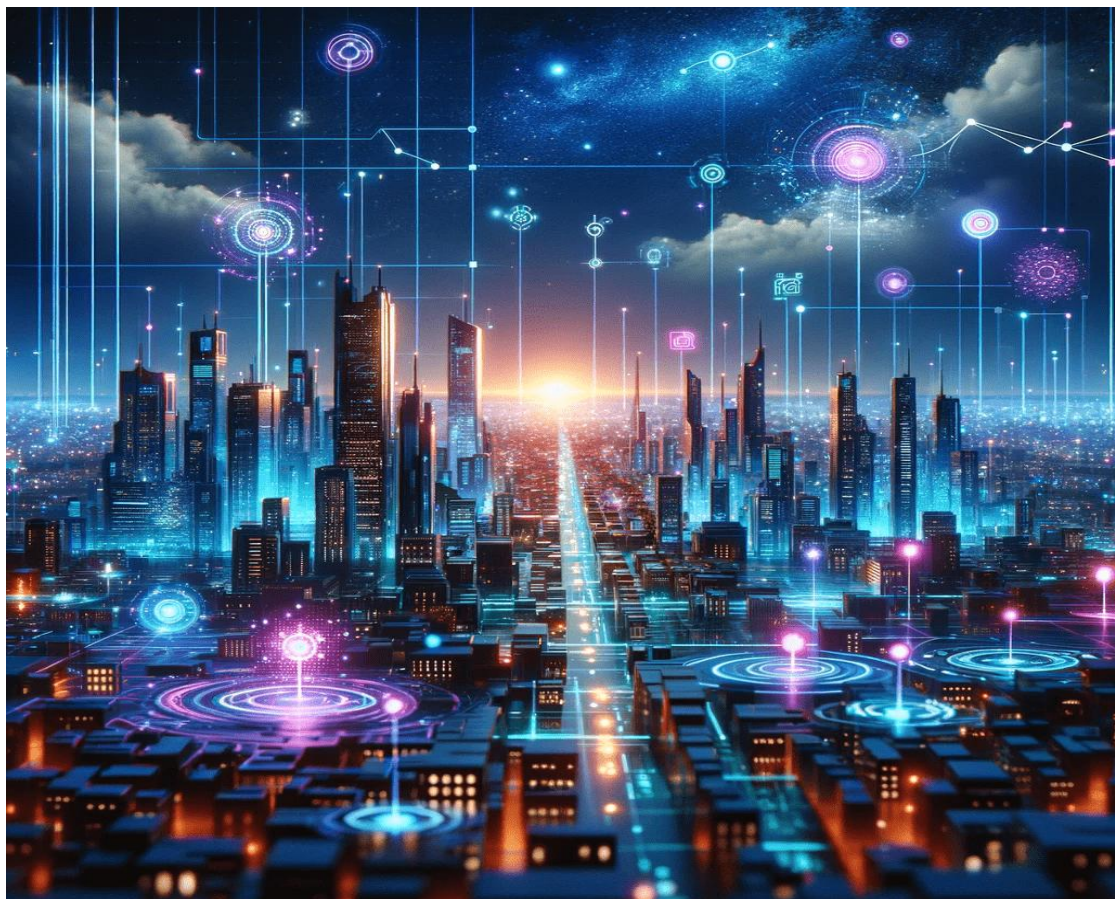


Rapport de
Projet de Programmation Orientée Objet

**Réaliser un outil de gestion de bibliothèque
d'images simplifié**



Réalisé par :

1. Ahamat Hisseine Youssouf
2. Moussa Diop

Plan du travail :

Introduction

I. Fonctionnalités du Programme

1. Application réduite

2. Les transformations d'images

3. Support des tags et sauvegarde des transformations

4. Sécurité

II. Conception du projet

III. Défis rencontrés

Conclusion

Introduction

Ce projet vise à développer un outil de gestion de bibliothèque d'images simplifié. Avec l'augmentation exponentielle des données visuelles dans notre environnement numérique, la gestion efficace des images devient cruciale pour de nombreuses applications allant de l'archivage numérique à la manipulation graphique en passant par la sécurité des données.

L'objectif de ce projet est de créer une application capable de gérer une collection d'images de manière intuitive et sécurisée. Pour atteindre cet objectif, l'outil intégrera des fonctionnalités essentielles telles que le traitement d'images, la gestion des tags pour une recherche facile, et des fonctions de chiffrement pour sécuriser les images stockées.

Ce rapport détaille le processus de conception, de développement et de mise en œuvre de l'outil, en mettant l'accent sur l'adoption du paradigme MVC (Modèle-Vue-Contrôleur) pour assurer une architecture logicielle claire et maintenable. Nous discuterons également des défis rencontrés lors du développement et des solutions adoptées pour surmonter ces obstacles, tout en fournissant une analyse critique de l'efficacité de l'outil développé.

I. Fonctionnalités du programme

Passons maintenant à une description détaillée des fonctionnalités développées pour notre outil de gestion de bibliothèque d'images, où nous examinerons les aspects techniques et les capacités spécifiques de l'application.

1. Application réduite

Le déroulement de l'application réduite dans votre projet JavaFX, basé sur l'architecture du projet et les extraits de code fournis, se présente comme suit :

1) Lancement de l'Application:

HelloApplication est la classe principale qui hérite de Application de JavaFX. Le programme commence ici.

La méthode start() est appelée, chargeant le fichier FXML du sélecteur de fichier (FileChooser.fxml) et affichant la fenêtre principale avec le titre "Gestion d'image".

2) Sélection de l'Image:

Lorsque l'utilisateur interagit avec l'interface, il clique sur le bouton géré par FileChooserController.

Le contrôleur ouvre un dialogue FileChooser, permettant à l'utilisateur de sélectionner une image à partir des filtres d'extension spécifiés (.png, .jpg, .gif).

3) Ouverture de la Fenêtre de Traitement d'Image:

Une fois l'image choisie, FileChooserController charge une nouvelle vue (traitement.fxml) et initialise TraitementController.

L'URL de l'image sélectionnée est récupérée et passée à TraitementController pour un traitement ultérieur.

4) Affichage de l'Image Sélectionnée:

TraitementController reçoit l'URL de l'image et l'utilise pour créer un objet Image, qui est ensuite passé à un ImageView pour l'affichage.

Une nouvelle scène avec le ImageView affichant l'image est mise en place, et la fenêtre de l'application est mise à jour pour montrer cette nouvelle scène.

5) Gestion des Ressources Images:

Toutes les images manipulables par l'application sont stockées dans le dossier ressources/images, ce qui offre une organisation logique et un accès facile aux images.

2. Les transformations d'image

Ces transformations sont des opérations fondamentales dans le traitement d'images numériques et permettent à l'utilisateur d'ajuster visuellement les images selon ses besoins.

1) Récupération de l'Image

Chaque transformation débute par la récupération de l'image source à l'aide de la classe Image du package javafx.scene.image. Cette classe encapsule les données d'une image chargée, qui peut être récupérée soit à partir d'un fichier, d'une URL ou d'une source de pixels.

2) Création d'une Image Modifiable

Un objet WritableImage est créé pour stocker le résultat des transformations appliquées à l'image source. Ce type d'image permet la manipulation de ses pixels grâce à un PixelWriter, qui est obtenu via la méthode getPixelWriter() de l'objet WritableImage.

3) Parcours et Manipulation des Pixels

Les pixels de l'image originale sont lus et traités en utilisant deux boucles for imbriquées qui parcourent l'image en largeur et en hauteur. PixelReader est utilisé pour accéder aux données des pixels de l'image source.

4) Application des Transformations

Pour chaque pixel, la transformation choisie est appliquée :

Rotation : La classe Rotation applique une rotation à un ImageView en ajustant sa propriété rotate, ce qui tourne l'affichage de l'image sans modifier les pixels eux-mêmes.

Symétrie Horizontale et Verticale : Les classes SymetrieHorizontal et SymetrieVertical implémentent l'interface Symetrie et réarrangent les pixels de l'image source pour créer un effet miroir en réfléchissant l'image autour de l'axe horizontal ou vertical.

5) Écriture des Pixels Transformés

Les pixels transformés sont ensuite écrits dans le nouvel objet WritableImage. Pour chaque pixel transformé, la méthode setArgb() du PixelWriter est utilisée, qui prend les valeurs ARGB calculées pour le nouveau pixel et les assigne à sa nouvelle position dans l'image résultante.

6) Intégration dans l'Interface Utilisateur

Les transformations sont intégrées dans l'interface utilisateur par le biais de TraitementController. Les méthodes de ce contrôleur sont liées aux actions de l'utilisateur (par exemple, les clics sur les boutons correspondant à chaque transformation), déclenchant ainsi le processus de transformation et la mise à jour de l'image affichée dans le ImageView.

3. Support des tags et sauvegarde des informations

Ajout de Tags : Les tags sont recueillis à partir d'un champ de saisie (inputTags) et convertis en une liste. Cette liste est ensuite assignée à l'objet ImageSaved via setTags.

Enregistrement des Images : Lors de l'enregistrement, l'objet ImageSaved est ajouté à une liste récupérée depuis le fichier JSON si l'image n'existe pas déjà, garantissant qu'aucun doublon n'est créé. Les nouveaux tags et l'identifiant unique sont également sauvegardés.

4. Sécurité

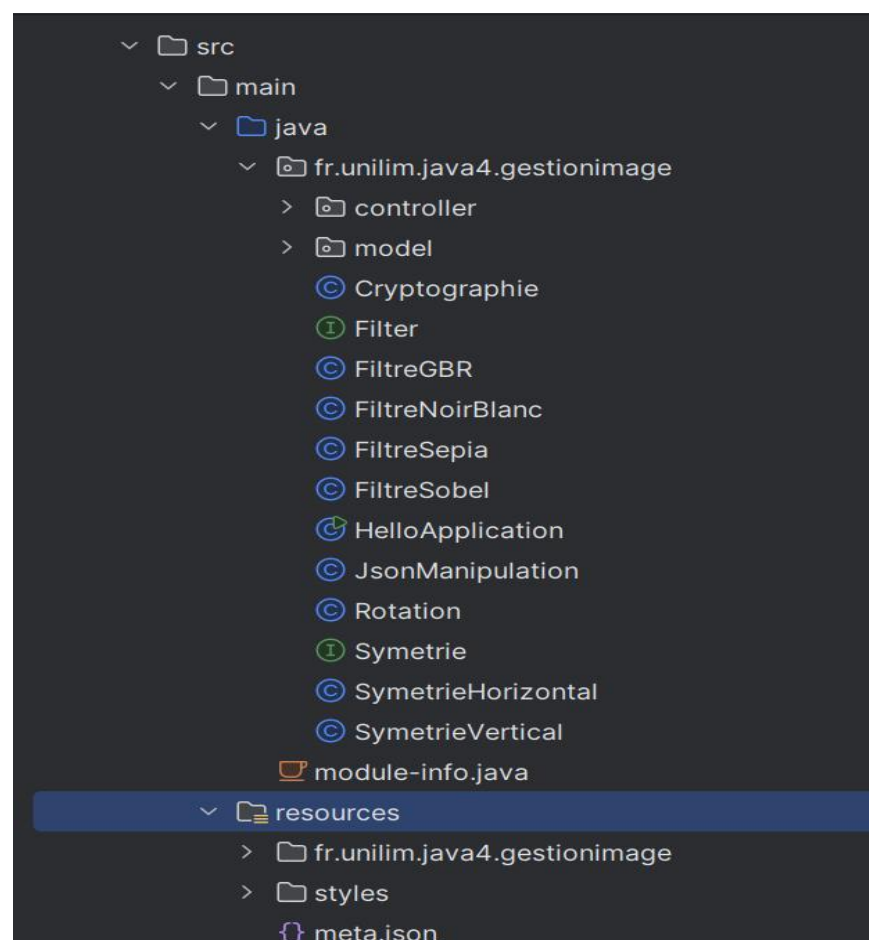
la méthode `cryptImage()` dans `TraitementController` démontre comment une image peut être cryptée:

Initialisation de l'Objet de Cryptographie : Une instance de la classe `Cryptographie` est créée en passant l'image actuelle affichée dans `imageView`.

Cryptage de l'Image : La méthode `encryptImage()` de l'objet `Cryptographie` est appelée avec le mot de passe fourni par l'utilisateur (`password.getText()`). Cette méthode est supposée utiliser `SecureRandom` pour mélanger les pixels de l'image en fonction du mot de passe, rendant l'image méconnaissable sans la connaissance du mot de passe exact.


II. Conception du projet


L'architecture de notre projet est représenté dans l'image suivante :



Lorsque le programme est lancé, une page permettant de sélectionner une image s'affiche. Une fois l'image choisie, elle est affichée sur une nouvelle page équipée de boutons offrant diverses fonctionnalités de base. L'utilisateur peut alors appliquer différentes transformations à l'image via ces boutons, les modifications étant instantanément visibles. Il est également possible de crypter l'image en saisissant un mot de passe et en validant cette action ; l'image cryptée sera alors affichée à côté de l'image originale. De plus, un champ de saisie est disponible pour que l'utilisateur

puisse ajouter des tags à l'image avant de valider. Enfin, un bouton "Enregistrer" permet de sauvegarder l'image avec toutes les modifications appliquées.

 Gestion d'image



Symetrie Vertical

Symetrie horizontal

Filter sobel

Filtre noir/blanc

Filtre sepia

Filtre GBR

Rotation

Enregistrer

Tags

Crypt

Search images

III. Défis rencontrés

Durant la réalisation de ce projet, plusieurs défis majeurs ont été rencontrés. L'un des plus significatifs a été la gestion efficace des transformations d'image en temps réel, ce qui demandait une optimisation constante des performances pour éviter les ralentissements lors de l'application des filtres et des modifications. Un autre défi était la sécurisation du processus de cryptage des images : il fallait garantir que le cryptage soit à la fois sûr et réversible uniquement par l'utilisateur ayant le mot de passe correct. Ces défis ont nécessité des recherches approfondies, des tests rigoureux et une série d'itérations de conception pour aboutir à des solutions efficaces.

Conclusion

En conclusion, ce projet de création d'un outil de gestion de bibliothèque d'images a non seulement permis de développer une application fonctionnelle répondant aux besoins spécifiques de manipulation et de sécurisation des images, mais a également offert une plateforme riche pour l'application des connaissances théoriques en programmation JavaFX et en cryptographie. Grâce à ce projet, des compétences essentielles en développement de logiciels, gestion de base de données, interface utilisateur et sécurité informatique ont été renforcées. Malgré les défis rencontrés, les solutions adoptées ont conduit à un outil robuste qui facilite la gestion et la transformation sécurisée des images. Ce projet sert non seulement de fondement solide pour des applications futures plus complexes, mais ouvre également la voie à des améliorations et extensions, telles que l'intégration de capacités de traitement d'image avancées et de fonctionnalités de réseau social pour le partage d'images.